Algovize

Snaha o vizualizaci intuice

Povídání pro studenty

Rozšiřujícího semináře ADS na MFF UK

prof. RNDr. Luděk Kučera, DrSc.
Algovision.org
a
MFF UK a FIT ČVUT

8. března 2021

1 Úvod

Zpříma a nahrubo se dá říci, že Algovize je soubor animací (nebo, chcete-li, vizualizací) algoritmů. To je obor, který zažil svá zlatá léta někdy v období 1995-2010 (možná pro slovo 'zlatá' by bylo vhodné to rozmezí na obou koncích zúžit).

Nyní již takových systémů přežívá jen několik, například VisuAlgo dr. Stevena Halima (National University of Singapore) a vizualizace dr. Davida Gallese (University of San Francisco). Cílem Algovize není kontinentální vyvažování, ale především využití grafických a vizuálních prostředků k předání myšlenek, které podle mého názoru jsou v jiných systémech vizualizací algoritmů nedostatečně rozvinuty.

Tento text je drobná kazuistika, popisující některé aspekty Algovize, ilustrující odlišný přístup k této oblasti. Uváděné příklady se dají zhruba rozdělit do dvou skupin:

- **Vizualizace intuice** je základ filosofie Algovize; dá se jen obtížně popsat jinak, než prostřednictvím příkladů - zhruba řečeno autor al-

goritmu byl k dosažení cíle veden svými představami, které obvykle dost přesahují to, co je pak napsáno v článcích a učebnicích. Snažím se autorovu intuici najít (to je to, co má člověk v hlavě, když po dlouhé době strávené nad článkem řekne "teď už tomu opravdu *rozumím*") a prostřednictvím vhodného zobrazení ji učinit zřejmou v nabízené vizualizaci.

- **Vizuální upozornění**: přednáší-li učitel jisté téma dostatečně dlouho, objeví i řadu možností, jak to téma *špatně pochopit* nebo typické způsoby, jak něco (třeba i dosti důležitého) ve výkladu nebo učebnici přehlédnout. A popíšu zde několik příkladů, jak vizuální prezentaci udělat tak, aby to, co se jindy přehlédne, z obrázku či animace přímo "trčelo" a aby bylo prakticky nemožné to přehlédnout.

- **Dobré pochopení látky pomocí hry s vizuálními objekty**

2 Vizualizace intuice

V textu "Odkud přišla Algovize" je popisován bouřlivý rozvoj animací algoritmů, jako pomůcky pro jejich výuku, na přelomu tisíciletí, i jejich úpadek mezi roky 2005-10.

Data si pro zobrazení například ve vědeckém článku nebo učebnici odedávna zobrazujeme graficky - třeba matice, grafy ve smyslu Excelu (sloupcové, výsečové, spojnicové a jiné) i grafy ve smyslu teorie grafů (s vrcholy znázorněnými kolečky a hranami jako jejich spojnicemi), obrázky geometrických útvarů v 2D i 3D a podobně.

Když v druhé polovině 90. let přišly programovací systémy, umožňující snadno programovat dynamickou grafiku (třeba Java s jejími knihovnami), zdálo se, že bude snadné vizuálně popsat práci libovolného algoritmu a že také bude snadné z té animované vizualizace ten algoritmus pochopit a naučit se ho.

Pak ale postupně přišlo rozčarování, pochopit z animace princip algoritmu se ukázalo jako test inteligence, jehož laťka byla nastavena vysoko nad schopnosti i těch nejlepších studentů (a učitelé na tom byli úplně stejně, ne-li hůř). Je o tom příklad se Super Bowlem, o kterém píši ve výše citovaném textu.

Situace před 10-15 lety byla skvěle vystižena názvem jednoho článku: Ronit Ben Bassat-Levy, Mordechai Ben-Ari: We work so hard and they don't

use it: acceptance of software tools bz teachers, ITiCSE'07, Proc. SIGCSE Conf. on Innovation and technology in computer science education, 2007, str. 246-250.

Domnívám se, že to bylo především tím, že animace algoritmů vytvářeli odborníci na počítačovou grafiku, kteří vzali učebnici algoritmů (nebo výklad od kolegy) jako zadání, které naprogramovali, tak jak to tam bylo napsáno, využitím výše popsaných principů. Pak ty animace dali učitelům - ukázalo se, že animace byly šikovné jako ilustrace poté, co studentům algoritmus vysvětlili jinak, ale jako přímý nástroj výuky moc užitečné nebyly.

Ono je to totiž tak, že když autor, který začal vymýšlet novou výpočetní metodu, měl v hlavě globální představu jakým směrem by se měl dát a pak tím směrem šel a měl-li dost vynalézavosti a vytrvalosti a štěstí, pak k tomu cíli došel (i když se třeba ukázalo, že ležel hodně jinde a nebo našel něco jiného, než hledal).

Jako když Kolumbus měl představu, že je třeba plout na západ, chce-li se dostat na východ, nebo teď jiní doufali, že postačí do ramene píchnout vhodnou mRNA a tělo si vakcinační antigen udělá samo.

Někdy je ta intuice v oboru obecně známá, ale jen jednomu se jí podaří využít, jindy bylo základem se na věci podívat úplně jinak než ostatní (a pak už to šlo relativně snadno).

Když je pak cíle dosaženo, trvá dost času celou cestu k němu postupně vyrovnat, rozšířit, vydláždit a nakonec se všechno přehledně sepíše do článku, který se napřed přednese na konferenci, pak otiskne v časopise a nakonec se třeba dostane i do učebnice. V tu dobu je výsledek už mrtvý, vypreparovaný jako anatomický preparát ve formalinu nebo vycpaný pták ve školním kabinetu a původní autorova představa, která ho vedla, se často úplně vytratí.

A pak k tomu preparátu v učebnici nebo ve skriptech přijde student, buď veden svým učitelem, nebo mnohdy odkázán sám na sebe, a má-li tomu algoritmu opravdu porozumět a ne se pouze naučit posloupnost kroků, o kterých jen tuší proč to vlastně funguje, je třeba aby se písmeny na stránkách a případnými statickými obrázky prokousal k průvodnímu pohledu autora algoritmu.

Jak bylo uvedeno výše, ukázalo se, že (na rozdíl od počátečních nadějí) klasická animace temporálních změn standardním způsobem zobrazených dat není v tomto snažení příliš velkou pomocí.

Při vytváření Algovize jsem se snažil o překlenout propast mezi představou tvůrce algoritmu a snahou studenta se k té představě dopracovat, tím, že tu propast, na jejímž dně leží ten text v učebnici, překlenu nějakým mostem,

který k tvůrcově intuici povede přímějším způsobem.

Obliba, kterou si Algovize získala prostřednictvím Rozšiřujícího semináře u studentů matfyzu, možná vychází z toho, že se mi ten cíl alespoň trošku daří dosahovat. A moc mne potěšilo, že když John Stasko z Georgia Inst. of Technology, který byl asi nejznámější postavou zlatého věku animací, a přijel předloni na konferenci do Průhonic, mi o dynamické vizualizaci Dijkstrova algoritmu v Algovizi řekl něco jako "kdybychom to tak tehdy dělali, nemuseli bychom s tím skončit".

To povídání je zatím jen abstraktní blouznění, ale pokusím se to ilustrovat na několika příkladech z Algovize, které ukazují, jak si představuji, že se obecná intuitivní představa dá někdy vizuálně zobrazit.

Moje vlastní zkušenost a podle rozhovorů s kolegy i často zkušenost jejich je, že odborník, pokoušející se najít novou výpočetní cestu, mnohdy uvažuje v obrázcích, kterými v duchu manipuluje a zkoumá je, a mojí snahou je ty obrázky nějak dostat na obrazovku a popsat Vám je. Snad se to alespoň někdy alespoň trochu podaří.

2.1 3D pohled na diagram Voroného

Animace algoritmu, který objevil Steve Fortune pro hledání diagramu Voroného, se dá najít na několika webových stránkách v podobě, podobné scéně "Animace" v kapitole "Geometrické algoritmy", procházka "Diagram Voroného". Jsou to klasické příklady animací, ze kterých není poznat, co a proč se tam děje, pokud už ten algoritmus neznáte.

A přitom je to velmi snadné ukázat, je-li k dispozici dynamická 3D grafika, viz scény "Kužely" a "Zametací rovina". A o tom, že cesta k cíli se stává postupně stále schůdnější, jsou poslední dvě scény té vizualizace, které ukazují, jak to vše původně Fortune viděl.

2.2 Statický a dynamický Dijkstrův algoritmus

V Algovizi jsou dvě vizualizace Dijkstrova algoritmu pro hledání nejkratší cesty v nezáporně hranově ohodnoceném grafu (kapitola "Grafové algoritmy", procházka "Extremální cesty").

"Statická" vizualizace má ten název proto, že vrcholy grafu v průběhu výpočtu zůstávají stále na stejném místě, mění se barvy a někdy i tvar hran a čísla, kterými jsou vrcholy a/nebo hrany popsány. Tuto animaci můžete najít na nesčetných stránkách na internetu, protože Dijkstrův algoritmus je

ve většině univerzitních informatických programů nejsložitější a závěrečný algoritmus, který je probírán v základní přednášce o algoritmech a datových strukturách.

Ta vizualizace hrubě porušuje základní pravidlo srozumitelné vizualizace: důležitá informace nesmí být předávána pouze ve formě čísel, které je třeba číst.

Když v prezentaci uvedete posloupnost 33-27-29-42-53-27, nikdo z posluchačů si nestihne udělat představu, jak posloupnost vlastně vypadá; raději ji tedy ve svém Excelu nebo PowerPointu znázorníte graficky, třeba sloupečky, a pak trend nebo jeho absenci vidíte ihned.

To důležité, co je v animaci třeba ukázat, je odhad vzdálenosti, který se pro každý vrchol v průběhu výpočtu stále mění. Dynamická animace algoritmu nechává vrcholy klouzat po vodorovných přímkách tak, že jejich x-ová souřadnice je lineárně závislá na odhadu vzdálenosti, který se snažím ukázat vizuálně srozumitelným způsobem.

A nejde jen o to, že pozorovatel animace má dobrou představu o hodnotách odhadů, aniž by musel luštit nenázorná čísla. To nejdůležitější je, že se najednou zcela zřetelně objeví ty zákonitosti, kvůli kterým algoritmus funguje a které jsou ve statické vizualizaci úplně ztraceny. Např.:

- Vrcholy, které již byly probrány a jsou uzavřeny, se nacházejí vlevo od jisté dělící linie a otevřené vrcholy jsou vpravo (okamžitě je vidět, že jakýkoli uzavřený vrchol má svůj odhad menší nebo rovný odhadu libovolného vrcholu otevřeného). A ještě lepší je tu "jistou" dělící linii natvrdo graficky znázornit, třeba jako hranici mezi tmavým a světlejším pozadím, pak už si ji všimne každý.

- Otevřený vrchol, který si Dijkstrův algoritmus vybere pro probrání a uzavření, je ten z otevřených, který má k té dělící linii nejblíže a je tedy nejvíc vlevo. Je až nefér, jak vizualizace *napovídá* a přímo pozorovateli tuto myšlenku vnutí, zatímco ve statickém zobrazení je docela pracné a nenázorné ten vrchol určit bez pomoci systému.

- Při relaxaci hrany probíraný vrchol konec hrany "přitahuje" k sobě, ten se ale nedostane nalevo od přitahujícího vrcholu, protože jsou ohodnocení hran *nezáporná*.

Už se nezjistí, zda se Edsger W. Dijkstra na graf díval tímto způsobem, ale lze si alespoň představit, že to tak bylo a s touto představou je činnost algoritmu celkem snadno představitelná a předvídatelná.

Pedagogický experiment, který jsem před časem udělal na FIT ČVUT (a ještě jsem si nedal práci s tím výsledky publikovat) ukázal, že pomocí dynamické animace studenti pochopili výrazně lépe a rychleji základní myšlenky, na kterých je algoritmus založen.

2.3 Binomická halda

Ví se, že Jean Vuillemin, který popsal binomickou haldu, na ni přišel na základě analogie s binárním sčítáním. Málokdy je tak snadné vizualizovat autorovu představu, jako je to zde (viz Algovize: "Datové struktury - binomická halda").

2.4 Dinitzův algoritmus

Potíž Ford-Fulkersonova algoritmu je v tom, že použitelné šipky, po kterých lze přenést přebytek, nejen ubývají, ale mnohé se zase vracejí, takže se algoritmus může dlouho "točit dokola". Algoritmus vylepšili současně a nezávisle na sobě Edmonds s Karpem a Dinitz. Yefim Dinitz svou implementací přímo popsal svůj pohled na data, takže je na první pohled vidět, že šipky, které se znovu objeví, směřují nejprve doleva, zatímco ty, které mohou mizet, jdou doprava. A vhodným způsobem algoritmus omezil tak, že doleva směřující šipky se nemohou používat. Pak už algoritmus směřuje velmi rychle ke svému cíli.

Grafický přístup Algovize Dinitzův přístup využívá přesně jak byl popsán; existuje málo jiných dostupných animací Dinitzova algoritmu, obvykle se takové kolekce omezují na jednoduchoučké algoritmy - a vizuální popis klíčové myšlenky tam vidět není ke škodě srozumitelnosti výkladu.

2.5 Korespondence mezi červeno-černými stromy a speciálními B-stromy (2-3-4-stromy)

V klasickém zobrazení červeno-černého stromu si černé vrcholy přitáhnou své červené syny a dostaneme zobrazení ve kterém je B-strom již vlastně vidět, a stačí klastry vrcholů překreslit formou, obvyklou u B-stromů a téma je vysvětleno během pěti minut.

I když to myslím historicky bylo opačně, postranní datové boxy B-stromů se spustily dolů a (jak autoři popisovali) se dostupnými fixy obarví jinak a vznikly č-č stromy.

A i mne překvapilo, jak pojmy a operace pro červeno-černé stromy se přímo překládají na pojmy a operace B-stromů, byť bez tohoto zobrazení by se zdálo, že jsou to zcela odlišné věci (například hloubka B-stromu a černá hloubka červeno-černého stromu nebo přesun barev v červeno-černém stromu přímo koresponduje s rozštěpením maximálního vrcholu B-stromu na dva minimální).

Korespondence je popsána v Algovizi v "Datové struktury:Červeno-černé stromy", scéna "Červeno-černý a 2-3-4".

2.6 Velikost stromů Fibonacciho haldy

Animace algoritmů jsou obvykle omezeny na jednoduchá až triviální algoritmy a Fibonacciho halda je již pokročilé téma, které je pokryto málo. Nenašel jsem vizualizaci Fibonacciho haldy, která by se snažila nějakým způsobem předat základní myšlenku: jestliže má strom haldy velký stupeň kořene, musí mít velmi velký počet vrcholů (a další krok ve výkladu je ukázat, že minimální počet vrcholů roste v závislosti na stupni vrcholů jako Fibonacciho řada - odtud název). Jedině s pochopením této závislosti je možné zjistit, že operace Fibonacciho haldy pracují nejhůře v logaritmickém (amortizovaném) čase.

Je-li F_k Fibonacciho strom s minimálním počtem vrcholů pro stupeň kořene k, pak je třeba studentům ukázat, že F_k se získá sloučením stromů F_{k-1} a F_{k-2}. To je sice zakázaná operace, ale dá se nasimulovat sloučením dvou F_{k-1} (povoleno) a maximálním možným "oškubáním" vzniklého stromu. Konstrukce se dá vizualizovat tak, že je vidět, že ta myšlenka je naprosto přirozená a jednoduchá.

2.7 Bitonické třídění

U pojmu bitonické posloupnosti dělá výklad nepřehledným to, že sice bitonická posloupnost v zásadě posloupnost jdoucí nahoru a pak dolu (bi tonus = dva směry změny), ale definice musí být uzavřena na rotace. Běžný popis posloupnosti lineárním způsobem (například v textu) si s rotací příliš nerozumí. Aby se ukázalo, že jde o jednoduchý pojem, je třeba použít cyklickou reprezentaci, ukázat její vztah s reprezentací lineární, a pak hlavní část výkladu provést pomocí cyklické podoby posloupnosti.

2.8 Metoda konjugovaných gradientů

Metoda konjugovaných gradientů (CGM) je základem pro numerickou lineární algebru (iterativní řešení velkých řídkých soustav lineárních rovnic). Její implementace nyní slouží jako alternativní benchmark pro hodnocení výkonu superpočítačů. Studenti informatiky se v kurzu lineární algebry obvykle myšlenku CGM *skoro* naučí, stačí dát pojmy, které si osvojili, vhodně dohromady.

Myšlenkou metody je transformovat pozitivně definitní symetrickou matici na jednotkovou, provést tam sestup ve směru nejprudšího gradientu a vrátit se zpět. A pro problém v dimenzi 2 se ta transformace snadno vizualizuje lineární deformací plochy, odpovídající kvadratické formě, odpovídající matici, a znázorněné pomocí 3D grafiky, tak, aby vrstevnice na ploše se z elips staly kružnicemi.

2.9 Knuth-Morris-Prattův algoritmus pro vyhledávání vzoru v textu

Ve scéně "Plovoucí vzor" je pomoci jednoduchého zobrazení přirozeně vysvětlen základní pojem stavu výpočtu jako délky nejdelšího prefixu vzoru, který je současně sufixem přečteného textu". Zobrazení ve scéně "Shodné prefixy" zase umožní ukázat současně všechny prefixy vzoru, mezi nimi zvýraznit shodné prefixy (které jsou sufixy známého textu) a přirozeným způsobem popsat, jak poznat, co budou shodné prefixy v následujícím kroku. Pak je již výklad samotného algoritmu vcelku triviální.

2.10 Simplexový algoritmus

V 2D i 3D je ukázáno, že soustava lineárních nerovností definuje mnohoúhelník nebo polyedr, na nichž hledáme extrémní bod v daném směru. To je sice interpretace dobře známá, ale ne vždy dostatečně vysvětlená, viz způsob, jak se Lineární Programování přednáší třeba na VŠE.

A dle mého názoru nestačí ukázat geometrický pohled, aniž by se také ukázalo, jak se od něj přejde k tomu formálně-matematickému s pivoty a lineárními manipulacemi. K obrázku mnohoúhelníka nebo polyedru se ještě dá seznam těch nerovností a ukazuje se, že pohybuje-li se bod mezi vrcholy tělesa po jeho hranách, je ve vrcholu N nerovností splněno jako rovnost (kde N je dimenze problému) a při pohybu podél hrany se jedna rovnost "uvolní"

a optimalizovaný funkcionál se nechá růst dokud se nenarazí na rovnost v další nerovnosti.

Až poté studenti mohou simplexovému algoritmu dobře rozumět a zároveň jej prakticky používat.

2.11 Vynechání klíče vrcholu se dvěma následníky v binárním vyhledávacím stromu

Tento příklad by se mohl nazvat prkotinou a asi také je, není to místo, které by studentům dělalo problém, ale myslím si, že to je také povedená ukázka využití vizuální představy. Klíč vrcholu se dvěma následníky v BVS nevynecháme tak, že bychom ho vyhodili i s vrcholem, ve kterém leží. Vyhodíme jen klíč, do slepého vrcholu se přesune klíč z jiného vrcholu, který půjde ze stromu vyhodit snáze.

Jde o to, odkud se ten klíč přesune. Ono to je trochu vidět, ale přece jen z obrázku stromu není zase tak snadno poznat jak tu operaci provést. A je to přitom úplně triviální, pokud si zapneme znázornění hodnot klíčů sloupečky pod vrcholy (volba "Ukaž hodnotu"). Do prázdného místa po vynechaném klíči se přesune sousední sloupeček a je to! A také je tam ihned vidět, že jiný než jeden ze dvou sousedních sloupců se tom přesunout nedá, má. A jediné, co je trochu složité, je nalezení vrcholu, jehož klíči odpovídá ten sousední sloupeček. To je ale již technická záležitost (vyřešená v předchozích scénách).

Tento velice jednoduchý příklad znovu ukazuje, jak je (obzvláště jedná-li se o příklad pro výuku) důležité nalézt úhel pohledu, ze kterého se zkoumaná situace najednou stane snadnou a přirozenou.

A věřím, že to je vždy ten pohled, kterým se na problém díval tvůrce algoritmu - prosekáváte-li se k cíli v neznámém terénu, těžko dojdete k cíli, máte-li komplikovaný a těžko uchopitelný plán. K cíli se dostane ten, jehož intuice a představy, které má v hlavě, ukazují přímo k cíli; i tak bývá překážek na cestě tolik, že je velmi snadné nedojít.

3 Vizuální důkazy

V této kapitole se vlastně dále rozebírají možnosti zobrazení intuice autora algoritmu. Důkaz správnosti algoritmu není nic jiného, než popis cesty, kterou autor k algoritmu došel. Cesta, kterou autor k algoritmu došel, zcela určitě

není jen formální manipulace symboly - takových možných manipulací je astronomické množství a bez konkrétní představy, jak by se k cíli mělo dojít, by se k němu autor nikdy nedostal. A ta představa může mnohdy být vizualizována a tím způsobem předána jako vodítko, co se vlastně ve formálně napsaném popisu algoritmu nebo důkazu jeho správnosti dělá.

3.1 Správnost 3D pohledu na Voroného diagram

Ve scéně "Kužely" v "Geometrické algoritmy:Diagram Voroného" je při vertikálním pohledu na kužely zřejmé, že viditelné části kuželů v projekci do roviny míst dávají hledané Voroného oblasti, ale na druhé straně je mnoho případů, kdy se výpočetní postup zdá správný, protože bývá někdy těžké vymyslet a představit si situaci, kdy postup nefunguje správně, byť taková situace existuje.

Proto Algovize nabízí i "vizuální důkaz": zvolený bod na povrchu některého kužele se doplní na trojúhelník přidáním průmětu bodu do roviny míst a vrcholu kužele. Je-li tento bod tam, kde se protínají dva nebo dokonce tři povrchy kuželů, dostaneme 2 nebo 3 trojúhelníky, které jsou v perspektivním pohledu na kuželové plochy zjevně shodné a ve vertikálním pohledu dávají dokazované tvrzení.

Formální důkaz správnosti Fortunova algoritmu není nic jiného, než formální popis právě popsané situace, takže na "vizuální důkaz" je možno hledět jako na snadno pochopitelný návod, jak číst formální důkaz (nebo pro studenta, kterému nedělá formální vyjadřování potíž, je to přímo návod, jak si formální důkaz sám napsat). Vizuální důkazy jsou proto velkým usnadněním a urychlením pochopení vykládané problematiky.

3.2 Korektnost rotací binárních vyhledávacích stromů

Klíče musí být do vrcholů binárního stromu vloženy jediným povoleným způsobem. Algovize ten způsob vizualizuje tak, že pod zobrazení každého vrcholu nakreslí sloupec s výškou úměrnou velikosti klíče. Podmínka binárního stromu pak je, že posloupnost sloupců roste zleva doprava.

Rotace jsou v Algovizi znázorněny tak, že vrcholy se při nich pohybují pouze ve *vertikálním* směru. To znamená, že systém sloupců se při rotaci vůbec nemění - a to je vizuální důkaz toho, že podmínku u umístění klíčů rotace neporuší (a současně je to i návod jak snadno napsat formální důkaz, nebo jak formální důkaz číst).

4 Vizuální upozornění

Dobrá vizualizace se také mnohdy uplatní tak, že obrázek velmi viditelných způsobem upozorní na fakta, kterých si čtenář textu mnohdy nevšimne a přitom jsou to někdy věci zcela principiální důležitosti.

Ukáži zde několik příkladů:

4.1 Rychlá Fourierova transformace

Když jsem zkoušel algoritmus rychlé Fourierovy transformace, přímo znepokojivé procento studentů mi ji přes usilovnou snahu a upozorňování při výkladu vykládalo takto:
Problém se rozdělí na dva podproblémy poloviční velikosti, ty se rekurzivně vyřeší a pak sloučí dohromady.

Psali přitom naprosto správně příslušný vzoreček, kde Σ nalevo představovalo původní problém a další dvě sigmy ty dva poloviční podproblémy

$$A_\ell = \sum_{k=0}^{N-1} a_k \omega_N^{k\ell} = \sum_{k=0}^{N/2-1} a_{2k} \omega_{N/2}^{k\ell} + \omega_N^\ell \sum_{k=0}^{N/2-1} a_{2k+1} \omega_{N/2}^{k\ell},$$

a jediné, co jim uniklo (taková drobnost, kterou se ani neobtěžovali zapsat - asi nejen na zkoušce, ale naneštěstí i předtím doma při přípravě), bylo, že vzorec pro původní problém byl pro $\ell = 0, \ldots, N-1$ zatímco ty podproblémy byly pro $\ell = 0, \ldots, N/2-1$ a o tom, že by je potřebovali i pro $\ell = N/2, \ldots, N-1$ ani nevěděli.

Z vizualizace (viz "Aritmetické algoritmy - Rychlá Fourierova transformace" ale je jasně vidět, že se původní problém rozdělí na *čtyři* podproblémy poloviční velikosti (dva pro $\ell = 0, \ldots, N/2-1$ a dva pro $\ell = N/2, \ldots, N-1$) - viz scény "Porovnání ... podmatic", jak to pozorný student pozná i z uvedených vzorců (ale nepozorný to přehlédne), protože se jedná o matice, kde počet prvků je druhou mocninou dimenze a to hlavní, na co James Cooley a John Tukey, autoři algoritmu FFT přišli, bylo, že jsou to dva páry shodných matic, viz scény "Překrytí ... podmatic", a teprve potom je to rozdělení problému na *dva* podproblémy poloviční velikosti.

Když jsem začal animaci rychlé Fourierovy transformace ukazovat na přednášce, procento těch, kteří opakovali výše uvedenou chybu spadlo prakticky na nulu, jeden z mých největších pedagogických úspěchů.

4.2 Vrcholy AVL stromů

Rozdíl výšek levého a pravého podstromu vrcholu AVL stromu nesmí být v absolutní hodnotě větší než 1. Tuto hodnotu často v algoritmech AVL stromů potřebujeme a proto ji Algovize znázorňuje tak, že vrcholy jsou jako vahadélka, která mohou být vyvážená nebo nakloněná nalevo či napravo. Přímý návod pro toto zobrazení jsem vzal z "mobilů" amerického sochaře Alexandra Caldera (1898-1976). Jeden z jeho mobilů (Big Red, 1959) je vyobrazen na obálce základní učebnice T. H. Cormen, Ch. E. Leiserson, R.L. Rivest, C. Stein: Introduction to Algorithms, 2nd ed., MIT Press, 2001 (Calder dělal také "stabily", ale ty Algovize využít neuměla).

Díval jsem se na animace AVL stromů na internetu, je jich hodně, je to jeden ze základních algoritmů, který se učí skoro všude. Všude jsem viděl vrcholy zobrazené kolečky, tu důležitou informaci o vyvážení vrcholů neukazují a rozdvojení, o kterém se píše v následujícím oddílu už by bylo úplně nemožné.

4.3 Balanční proměnné vrcholů AVL stromů

Když se do AVL stromu přidá nový vrchol tak, že strom stále splňuje AVL podmínku, nepozorný čtenář by si mohl myslet, že operace je zdárně ukončena.

Avšak pro rychlé provádění operací a rozhodování o případném vyvažování si do každého vrcholu dáváme proměnnou, uchovávající informaci o rozdílu výšek levého a pravého podstromu (s hodnotami -1, 0 a +1). Přidáním nového vrcholu se mohou stát hodnoty v mnoha vrcholech neaktuálními a je potřeba je upravit.

Algovize tuto nutnost připomíná tím, že po přidání nebo ubrání vrcholu se vrcholy s neaktuálními balančními proměnnými rozdvojí - naklonění přední kopie vrcholu znázorňuje skutečný stav, zadní kopie vyjadřuje neaktuální hodnotu balanční proměnné a pokud vyčnívá zpoza přední kopie, připomíná nám to, že ji bude nutné aktualizovat.

5 Hry s vizuálními objekty

Ve skutečnosti nejde o hraní, ale o to, aby se student dobře seznámil s některými operacemi, které jsou důležité pro dobré pochopení konkrétních algoritmů:

5.1 Rotace binárních stromů

Vyvažované binární stromy, například AVL-stromy a červeno-černé stromy, naprosto zásadním způsobem používají operaci rotace hrany stromu. Při obvyklém výkladu se operace zadefinuje a potom vzápětí konkrétně použije v zápisu algoritmu a v nejlepším případě se dokáže, že v tomto použití zaručí, že algoritmus vykoná to, co vykonat má.

Pokládám ale za důležité, aby studenti zjistili, jak silná je to operace a co všechno se s ní dá provést. Proto Algovize přináší scény, ve kterých si student může s rotacemi hrát, viz posledních 5 scén v "Datové struktury:Binární vyhledávací strom".

Průvodce Algovizí ve scéně "Rotace ve stromu" ukáže, jak se z libovolného stromu dostane strom minimální možné hloubky nebo naopak maximální možné hloubky a scéna "Rotační cvičení" vyžaduje po studentovi, aby z daného náhodně vytvořeného stromu rotacemi vytvořil jiný strom, který je na pozadí.

5.2 Vytváření AVL-stromů

Studenty oceňovaná scéna ukazuje na začátku obecný strom s výrazně označenými vrcholy, které porušují AVL-podmínku. Cílem je rotacemi z něj vytvořit korektní AVL-strom. Cvičením se student sám naučí hodně z toho, co je pak třeba při návrhu AVL-algoritmů.

5.3 Fibonacciho stromy

V "Datové struktury:Fibonacciho halda" má student v druhé scéně ("Fibonacciho stromy") možnost si sám konstruovat stromy Fibonacciho haldy a v následující scéně dostává úkoly - stromy nakreslené na pozadí a má vytvořit jejich kopii. Je to důležité cvičení, protože ten, kdo zná standardní a binomickou haldu si těžko umí představit, jak široká je třída přípustných stromů a na druhé straně si "ohmatat" omezení, které Fibonacciho halda přináší.

6 Závěr

Tento text bude rozšiřován, použití grafiky pro zviditelnění někdy dosti složitých myšlenek a postřehů autorů algoritmů je v Algovizi hodně, zatím jsou uvedeny ty, které mne hned napadly a našel jsem si čas je zde napsat.